PATENT

I hereby certify that on the date specified below, this correspondence is being deposited with the United States Postal Service as first-class mail in an envelope addressed to Mail Stop Appeal Brief –Patents, Commissioner for Patents, P.O. Box 1450. Alexandria, VA 22313-1450.

Feb 6, 2006
Date

Phoebe E Pegson
~~Denise Sheridan~~ Phoebe E Pegson

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 09/654,115 Confirmation No. : 5616

Applicant : Dean A. Klein

Filed : August 30, 2000 Attorney Docket No.

Art Unit : 2193 Customer No. : 27,076

Examiner : Tuan A. Vu

Title : SYSTEM AND METHOD FOR DETERMINING THE CACHEABILITY OF CODE AT THE TIME OF COMPILING

APPEAL BRIEF TRANSMITTAL

Mail Stop Appeal Brief –Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Transmitted herewith, in triplicate, is the Appeal Brief in this application, with respect to the Notice of Appeal filed on December 6, 2005. Enclosed is our check in the amount of $500 for filing this Appeal Brief.

Any deficiency or overpayment should be charged or credited to Deposit Account No. 50-1266. This transmittal is being submitted in duplicate.

Respectfully submitted,

DORSEY & WHITNEY LLP

Edward W Bulchis

Edward W. Bulchis
Registration No. 26,847

EWB:dms

Enclosures:
- Postcard
- Check
- Appeal Brief (+ 2 copies)
- Copy of this Transmittal

1420 Fifth Avenue, Suite 3400
Seattle, WA 98101
Tel: (206) 903-8800
Fax: (206) 903-8820

H:\IP\Clients\Micron Technology\00\500050.01\500050.01 appeal brief trans.doc

IFW
AF

Effective on 12/08/04

FEE TRANSMITTAL SHEET (FY 2005)

☐ Applicant claims small entity status (see 37 C.F.R. 1.27)

Complete if Known	
Application No.	09/654,115
Filing Date	August 30, 2000
First Inventor	Dean A. Klein
Group Art Unit	2193
Examiner Name	Tuan A. Vu
Atty. Docket Number	500050.01

METHOD OF PAYMENT (Check One)

☒ The Director is hereby authorized to charge any additional fee required under 37 C.F.R. §§ 1.16 and 1.17 and 1.136(a)(3) and credit any over payments to Deposit Account No. **50-1266;** Deposit Account Name: **DORSEY & WHITNEY LLP.**

☒ Check Enclosed.

Extra Claim Fees

Current Claims		Prior		Extra		Fee		Fee Paid
Total	-	20	=		x	$	=	$
Ind.	-	3	=		x	$	=	$
Multiple Dependent Claims					x	$	=	$
				Subtotal (Extra Claims)				**$**

Petition Fee Under 37 CFR 1.17(f), (g), & (h)

Enclosed is a Petition filed under 37 CFR as indicated below:

☐ Petition Fee under 37 CFR 1.17(f) **Fee $400**

§ 1.53(e)	to accord a filing date.
§ 1.57(a)	to accord a filing date.
§ 1.182	for decision on a question not provided for.
§ 1.183	to suspend the rules.
§ 1.378(e)	for reconsideration of decision on petition refusing delayed payment of maintenance fee in expired patent.
§ 1.174(b)	to accord a filing date to an application under §1.740 for extension of patent term.

☐ Petition Fee under 37 CFR 1.17(g) **Fee $200**

§ 1.12	**for access to an assignment record.**
§ 1.14	**for access to an application.**
§ 1.47	**for filing by other than all inventors or person not the inventor.**
§ 1.59	**for expungement of information.**
§ 1.103(a)	**to suspend action in an application.**
§ 1.136(b)	**for review of a request for ext. of time when** §1.136(a) not avail.
§ 1.295	for review of refusal to publish a statutory invention registration.
§ 1.296	to withdraw a req. for pub. after notice of intent to publish issued.
§ 1.377	for review of decision refusing to accept a maintenance fee filed prior to expiration of a patent.
§ 1.550(c)	**for request for ext. of time in** *ex parte* **reexam. proceedings.**
§ 1.956	**for request for ext. of time in** *ex parte* **reexam. proceedings.**
§ 5.12	for expedited handling of foreign filing license.
§ 5.15	for changing the scope of a license.
§ 1.5.25	for retroactive license.

☐ Petition Fee under 37 CFR 1.17(h) **Fee $130**

§ 1.19(g)	to request documents in a form other than provided in this part.
§ 1.84	for accepting color drawings or photographs.
§ 1.91	for entry of a model or exhibit.
§ 1.102(d)	to make an application special.
§ 1.138(c)	to expressly abandon an application to avoid publication.
§ 1.313	to withdraw an application from issue.
§ 1.314	to defer issuance of a patent.

FEE CALCULATION (Continued)

3. ADDITIONAL FEES

Large Entity Fee	Small Entity Fee	Fee Description	Fee paid
50	25	Surcharge - late provisional filing fee or cover sheet	$
130	65	Surcharge – Late nonprovisional filing fee or oath	$
180	180	Submission of IDS	$
40	40	Recording each patent assignment per property (times number of properties)	$
120	60	Extension for reply within first month	$
450	225	Extension for reply within second month	$
1,020	510	Extension for reply within third month	$
1,590	795	Extension for reply within fourth month	$
2,160	1,080	Extension for reply within fifth month	$
790	395	Submission After Final 1.129	$
500	250	Notice of Appeal	$
500	250	Filing a brief in support of an appeal	$500
1,000	500	Request for oral hearing	$
130	65	Terminal Disclaimer Fee	$
800	400	Design Issue Fee	$
790	395	Request for Continued Examination (RCE)	$
130		Request for voluntary publication or republication	$
500	250	Petition to Revive – unavoidable	$
1,500	750	Petition to Revive – unintentional	$
200		Filing for patent term adjustment	$
400		Request for reinstatement of term reduced	$
1,120		Extension of term of patent	$
OTHER FEE (specify)			$
		Subtotal (Additional Fees)	$500
		Total Amount of Payment:	**$500**

Submitted by:

CUSTOMER NUMBER **27,076**	DORSEY & WHITNEY LLP	1420 Fifth Avenue, Suite 3400 Seattle, WA 98101-4010 (206) 903-8800 *phone* / (206) 903-8820 *fax*

Name: Edward W. Bulchis

Reg. No.: 26,847

Signature: Edward W Bulchis

Date: Feb. 6, 2006

PATENT

I hereby certify that on the date specified below, this correspondence is being deposited with the United States Postal Service as first-class mail in an envelope addressed to Mail Stop Appeal Brief –Patents, Commissioner for Patents, P.O. Box 1450. Alexandria, VA 22313-1450.

Feb 6, 2006
Date

Phoebe E. Pogson
~~Denise Sheridan~~ Phoebe E. Pogson

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 09/654,115 Confirmation No. : 5616
Applicant : Dean A. Klein
Filed : August 30, 2000 Attorney Docket No.
Art Unit : 2193 Customer No. : 27,076
Examiner : Tuan A. Vu
Title : SYSTEM AND METHOD FOR DETERMINING THE CACHEABILITY OF CODE AT THE TIME OF COMPILING

Mail Stop Appeal Brief –Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF (37 C.F.R. § 41.37)

Sir:

This brief is in furtherance of the Notice of Appeal, filed in this application on December 6, 2005. The fees required under Section 41.20(b)(2) and any required request for extension of time for filing this brief and fees therefor are dealt with in the accompanying transmittal letter.

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is the assignee of this application, Micron Technology, Inc., a Delaware Corporation having a principal place of business at Boise, Idaho.

II. RELATED APPEALS AND INTERFERENCES (IF ANY)

There are no other appeals or interferences known to appellant, the appellant's legal representative, or the assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

1. Total Number Of Claims In Application

Claims in the application are: 1-61

2 Status Of All The Claims

A. Claims cancelled: none

B. Claims withdrawn from consideration but not cancelled: none

C. Claims objected to: none

D. Claims allowed or confirmed: none

E. Claims rejected: 1-61

3. Claims On Appeal

The claims on appeal are: 1-61

IV. STATUS OF AMENDMENTS

No amendment to the claims has been filed subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

1. Introduction

The claimed subject matter is directed to systems and methods for improving the efficiency of executing instructions by a processor. Various approaches have been developed to allow processors to execute instructions more efficiently. These approaches include branch prediction, instruction re-ordering, and caching. Branch prediction is used in a system where a processor pre-fetch instructions so they will be available to the processor as soon as the time comes to execute them. In branch prediction, the processor attempts to predict which branch of a program will be taken, and to pre-fetch instructions from the selected branch before the branch instruction is actually executed and the decision made to follow that or another branch. In instruction re-ordering, the order of the instructions in a program is changed so the instructions can be executed more efficiently. In instruction caching, all of the instructions in a program are stored in a first memory, which is normally system memory. A subset of the instructions, such as the most recently executed instructions or most frequency executed instructions, are stored in a second memory, known as a cache memory, that can be accessed more quickly than the first memory.

The claimed invention relates to instruction caching rather than branch prediction or instruction re-ordering. More specifically, rather than relying on the conventional approach of caching instructions based on either how frequently or how recently an instruction was executed, the disclosed caching system and method determines the cacheability of an instruction at compilation based on other factors, some of which are described in the specification. Based on this cacheability determination during compilation, the instructions are marked with at least one bit corresponding to the determination. During execution of the instructions, the cacheability bit is detected and used by a computer system to determine whether or not to cache the instruction. If the cacheability bit is detected during execution, the instruction is stored in cache memory after execution so that it can be more quickly fetched from the faster memory if it is necessary to subsequently execute the instruction. If the cacheability bit is not detected during execution, the instruction is not cached, and it must therefore be fetched from the slower memory if it is necessary to subsequently execute the instruction. The manner in which instruction caching occurs as specified in the independent claims will be described in greater detail below, it being understood that the specific references to the examples disclosed in the specification do not limit the claims to the disclosed examples.

2. Claim 1

Claim 1 is directed to a computer system 100 (Figure 1), which, as described on page 4 of the specification between line 6 and line 15, has cache circuitry 108 including a cache memory 110, 112. The computer system 100 also includes a main memory 118 and a processor 102 controlled by a computer program 200 (Figure 2), which is described in the specification between page 4, line 28 and page 5, line 9. Both the cache memory 110, 112 and the main memory 118 store information related to the computer program. The processor 102 is adapted to be controlled by the computer program 200 (Figure 2) and it directs 210 (Figure 2) selected portions of the information to the cache circuitry 108 based at least in part on cacheability determinations 202 (Figure 2) made during compilation of the computer program. Figure 4 shows a more specific example of a computer program that performs this function, which is described in the specification between page 8, line 16 and page 9, line 2. Finally, the computer system 100 includes bus circuitry in the form of bus interface units 104, 107, internal bus 105, CPU bus 106, and system controller 114 operatively connecting the processor 102, the cache circuitry 108 and the main memory 118.

In summary, during execution, the computer system of claim 1 directs selected portions of information to the cache circuitry 108 based on cacheability determinations made during compilation of the computer program.

3. Claim 25

Claim 25 is directed to a system 100 for determining which portions of a program code to cache and which not to cache. As described on page 4 of the specification between line 6 and line 15, the claimed system 100 includes a processor 102 connected to a memory device 118 containing a program code 200 (Figure 2). The program code 200 is described in the specification between page 4, line 28 and page 5, line 9. The processor 102 is adapted to be controlled by the program code 200 (Figure 2) to direct 210 (Figure 2) selected portions of the program code 200 to a cache 108 based at least in part on cacheability determinations 202 (Figure 2) made during compilation of the computer program 200.

Thus, the system 100 of claim 25, in a manner similar to the system of claim 1, directs selected portions of program code to the cache circuitry 108 based on cacheability determinations made during compilation of the computer program. Significantly, the cache 108 to which the information is directed includes a memory 110, 112 that is different from the memory device 118 initially containing the program code 200.

4. Claim 34

Claim 34 is directed to a method 200 (Figure 2) of controlling the cacheability of information in a computer system 100 (Figure 1) that includes cache circuitry 108. The method 200 includes both compiling a computer program and executing the computer program on the computer system 100. As described in the specification between page 4, line 28 and page 5, line

9, during compilation cacheability determinations are made 202 (Figure 2) for information associated with the computer program. Additionally, at least selected portions of the information are marked 204 (Figure 2) according to the cacheability determinations 202.

During execution, the marking of the selected portions of the information are detected 208 (Figure 2). The selected portions of the information are then directed 210 (Figure 2) to the cache circuitry 108 according to the marking.

5. Claim 48

Claim 48 is directed to a method 200 (Figure 2) for compiling a computer program. As described in the specification between page 4, line 28 and page 5, the claimed method 200 includes making cacheability determinations for information associated with a computer program 202 (Figure 2), and then marking 204 (Figure 2) at least selected portions of the information according to the determinations.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to be reviewed on appeal are:

(1) whether claims 25, 34 and 48, as well as the claims dependent thereon, are anticipated by U.S. Patent No. 6,115,809 to Mattson, Jr. under 35 U.S.C. § 102(e);

(2) whether claim 1, as well as the claims dependent thereon, would have been obvious under 35 U.S.C. § 103(a) over U.S. Patent No. 6,115,809 to Mattson, Jr. and in view of U.S. Patent No. 5,765,037 to Morrison.

VII. ARGUMENTS

1. Claims 25 Is Not Anticipated By U.S. Patent No. 6,115,809 to Mattson, Jr.

A. The Subject Matter Of Claim 25

Claim 25 reads as follows:

> 25. A system for determining which portions of a program code to cache and which to not cache, comprising:
>
> a memory device containing a program code; and
>
> a processor connected to the memory device, the processor being adapted to be controlled by the program code to direct selected portions of the program code to a cache based at least in part on cacheability determinations made during compilation of a computer program.

It should be noted that the system of claim 25 includes two instruction storage devices, a "memory device" and a "cache." The memory device stores a program, and the processor directs selected portions of the program code to the separate cache.

B. The Subject Matter Disclosed in the Mattson, Jr. Patent

The Mattson, Jr. patent discloses a method of controlling the manner in which branch instructions are processed. As is well-known in the art, branch prediction is desirable in a computer system that prefetches instructions for execution. Branch prediction is desirable because, when a branch instruction is encountered by prefetching circuitry in a processor, the prefetching circuitry does not know which branch should be followed for fetching additional instructions because the processor has not yet executed the branch instruction. In branch prediction, prefetching circuitry in the processor predicts which branch will be taken and then prefetches the instructions in that branch. In the method disclosed in the Mattson, Jr. patent, the branching behaviors of branch instructions in a computer program are first determined by tracing the instructions during execution of the instructions. (*See, e.g.*, "The present invention first profiles branch instructions within a trace to record branching behavior." Abstract). The branch instructions detected in the trace are divided into two groups. A first group consists of branch instructions having "strong" branching behavior in which the same branch was taken at least 80% of the time. A second group consists of branch instructions having "weak" branching behavior in which the same branch was taken less than 60% of the time. [Abstract]. "Branch instructions that are profiled to have 'strong' branching behavior...are placed in the group of branch instructions that are statically predicted. Branch instructions that are profiled to have 'weak' branching behavior...are placed in the group of branch instructions that are dynamically predicted." [Abstract]. The statically predicted groups of branch instructions are stored in one page of physical memory, and the dynamically predicted groups of branch instructions are stored in another page of the same physically memory. These separate pages of the same physical memory are referred to in the Mattson, Jr. patent as "static code caches" and "dynamic code caches." However, the use of the term "cache" in this manner has nothing to do with a caching system as that term has been used by applicant to describe a system in which cached instructions are stored in a memory that is separate from the main or system memory. During execution, a processor fetches the instructions from the memory and executes instructions using different algorithms depending on which page of memory stored the instruction.

The fact that the "static code cache" and the "dynamic code cache" are simply different pages in the same physical memory is apparent from the teachings in a number of locations in the Mattson, Jr. patent. For example, the cover which is a copy of Figure 5, shows the static code cache 38 and the dynamic code cache 40 as being separate "physical memory pages." As is well known to one skilled in the art, the term "pages" is typically used to refer to different rows or other contiguous locations of a memory. Similarly, the Mattson, Jr. patent states between line 55 of column 5 and line 3 of column 8:

> Prediction strategy flags 42 and 44 represent the branch prediction flags associated with the memory pages that comprise static code cache 38 and dynamic code cache 40, respectively. As shown in FIG. 2, certain prior art computer systems manufactured by the Hewlett-Packard Company include a branch prediction strategy flag in each translation lookaside buffer (TLB). The branch prediction strategy flag in each TLB determines the branch prediction strategy of branch instructions residing in the physical memory pages associated with the TLB. Accordingly, in such a computer system, all the branch prediction

> strategy flags in all the TLBs associated with the physical memory pages that comprise static code cache 38 are set to "static", and all branch prediction strategy flags in all the TLBs associated with the physical memory pages that comprise dynamic code cache 40 are set to "dynamic".

What is described in this excerpt is simply storing the strong and weak branch instructions in different pages of the same memory, which are identified by prediction flags set to "static" or "dynamic," respectively, in translation lookaside buffers associated with the different pages of a memory. Certainly it cannot be said that the Mattson, Jr. patent discloses that the "static code cache" is stored in one memory device and the "dynamic code cache" is stored in a different memory device, nor has the Examiner even attempted to establish that the Mattson, Jr. patent contains such disclosure.

C. *Summary Of The Rejection*

The final rejection dated June 6, 2005 rejects claim 25 as being anticipated by the patent to Mattson, Jr. However, the rejection does not specify a reference numeral of any component shown in the drawings of the Mattson, Jr. patent as corresponding to any limitation in claim 25. Instead, the final rejection relies on column 5, lines 25-56 and column 3, lines 46-51 of the Mattson, Jr. patent for disclosing a memory device containing a program code. Line 25-56 of column 5 relates to Figure 1, which is described as being "a trace from the prior art program code having a series of basic code blocks." [Col. 5, lines 6-7]. Thus, this portion of the Mattson, Jr. patent relied on for as disclosing the claimed memory device containing a program code does not describe any hardware device, and it certainly does not describe a memory device or even mention the word "memory."

The final rejection also relies on column 3, lines 46-51 for disclosing the claimed memory device containing a program code. However, this portion of the Mattson, Jr. patent simply references a prior art U.S. Patent No. 5,721,893. Although the rejection of claim 25 is based anticipated by the Mattson, Jr., patent, the Examiner may be attempting to combine the teachings of Mattson, Jr. with the teachings of the Holler patent. If so, an anticipation rejection cannot be based on a combination of references. Furthermore, the Holler patent is no more pertinent than the Mattson, Jr. patent. The Holler patent discloses a system in which branch instructions having matching predicted directions are grouped into common "buckets" in an array of buckets. The array of buckets is described as "the location in the branch prediction cache into which branch instructions are placed by the system hardware." [Col. 3, lines 38-41]. The Holler patent does not state or even suggest that the branch prediction cache is other than different locations of the same storage device. Thus, the Holler patent, like the Mattson, Jr. patent, does not describe different instructions being placed into different storage devices depending on whether or not they are cached.

For the claimed processor connected to the memory device that directs selected portion of the program code to a cache, the Examiner relies on the static code cache and dynamic code cache shown in Figure 3. However, Figure 3 is "a flowchart how branch prediction strategy may be varied for branch instructions in accordance with the present invention." [Col. 5, lines 12-14]. It does not disclose a processor or a memory device, and certainly not a processor directing selected instructions stored in one memory device to a cache.

Finally, the Examiner has not cited any subject matter of the Mattson, Jr. patent for the claimed operation of the processor to base its caching determinations on cacheability determinations made during compilation of a computer program. Instead, the Examiner has cited column 6, lines 18-45 and lines 59-62 of the Holler patent for this teaching. However, this portion of the Holler patent does not relate to making cacheability determinations. Instead, it describes putting different branch instructions in different buckets of the branch prediction cache based on their similarity of branching directions. The Holler patent does not describe or even suggest that different branch instructions should be put in different storage devices or that the decision about which bucket should receive which branch instructions should be based on cacheability determinations made during compilation of a computer program.

D. The Mattson, Jr. Patent Does Not Disclose All Of The Limitations Of Claim 25

As explained above, claim 25 includes a processor connected to a memory device containing a program code. The processor is controlled by the program code "to direct selected portions of the program code to a cache based at least in part on cacheability determinations made during compilation of a computer program." Thus, as explained above, claim 25 includes two instruction storage devices, a "memory device" and a "cache." The memory device stores a program, and the processor directs selected portions of the program code to the separate cache. As also explained above, the system disclosed in the Mattson, Jr. patent stores all of the branch instructions in different pages of the *same* physical memory. Similarly, the system disclosed in the Holler patent stores all of the branch instructions in different "buckets" of the same branch prediction cache. For this reason, neither the Mattson, Jr. patent nor the Holler patent discloses a system having a memory device storing a program, and a processor directing selected portions of the program to the separate cache. Therefore, neither of these references can be considered to anticipate claim 25.

As also explained above, the Mattson, Jr. patent is directed to branch prediction determinations; it does not describe or suggest making cacheability determinations during compilation or using the cacheability determinations to control caching during execution of the instructions. It is important to understand that a caching instructions is distinctly different from branch prediction. Some of the common differences between a cache system and a branch prediction system can be summarized as follows:

- In a caching system, a choice is made about which instructions are to be stored in the cache memory and which instructions are to be stored in main memory or some other memory that is separate from the cache memory; in a branch prediction system, all of the instructions in the predicted branch are generally stored in the same register or memory, as in the cited references.
- Branch prediction systems are used only when branch instructions are involved; cache systems are used to store instructions other than branch instructions, although they may also store branch instructions.
- Branch prediction systems are used only in pipelined processing systems in which instructions are prefetched; cache systems can be used in both processing systems where prefetching does not occur and processing systems where prefetching does occur.
- Branch prediction and caching solve different problems; Branch prediction solves the problem of allowing prefetching of instructions to

occur even though a branch has been encountered; caching solves the problem of the slow access times typically associated with main memory or other slow memory device. In fact, the slow access time of main memory is generally not an issue in prefetching system because the instruction has been prefetched by the processor by the time the processor is ready to execute it.

Therefore, while Mattson, Jr. and Holler may disclose an efficient means of storing branch instructions in different locations of the same physical memory, they do not disclose an efficient means of caching program instruction stored in a memory device by directing portions of the instructions stored in a memory device to a cache based on cacheability determinations made during compilation of a computer program.

For all of the reasons explained above, the anticipation rejection of claim 25, as well as the claims dependent thereon, on the basis of the Mattson, Jr. patent should be revered.

2. Claim 34 Is Not Anticipated By U.S. Patent No. 6,115,809 to Mattson, Jr.

A. The Subject Matter Of Claim 34

Claim 34 reads as follows:

> 34. A method for controlling the cacheability of information in a computer system, comprising:
>
> compiling a computer program, by:
>
> making cacheability determinations for information associated with the computer program; and
>
> marking at least selected portions of the information according to the determinations;
>
> executing the computer program on a computer system, the computer system including cache circuitry;
>
> detecting the marking of the selected portions of the information during execution of the computer program; and

directing the selected portions of the information to the cache circuitry according to the marking.

Claim 34 therefore includes both compiling and execution. During compilation, cacheability determinations are made for information associated with the computer program, and selected portions of the information are marked according the these determinations. During execution, the marking is detected and used by the computer system to direct the selected portions of the information to the computer system's cache circuitry.

B. The Subject Matter Disclosed in the Mattson, Jr. Patent

The subject matter of the Mattson, Jr. patent is described in detail above. In the interest of brevity, that explanation will not be repeated here.

C. Summary Of The Rejection

The final rejection dated June 6, 2005 also rejects claim 34 as being anticipated by the patent to Mattson, Jr. The claimed compiling a computer program by making cacheability determinations for information associated with the computer program was considered to be met by Figure 3 and column 6, lines 18-33 of the Mattson, Jr. patent or Figure 4 of the Holler patent, both of which describe making branch predictions as described above. The step of marking the selected portions of the information that are to be cached was considered to be met by the teaching in the Mattson, Jr. patent of marking branch instructions as either strong or weak, which is argued to be "equivalent to marking for appropriate cache storage action." The Examiner did not explain provide any legal authority for making an anticipation rejection based on this asserted "equivalence." Finally, the execution steps of detecting the marking of the selected portions of information and directing the selected portions of the information to the cache circuitry according to the marking were considered to be met by Figures 3 and 4 and column 9, lines 1-22 and 27-51 of the Mattson, Jr. patent. These portions of the Mattson, Jr. patent simply describe setting prediction flags for blocks of instructions following a branch as either "strong" or "weak" and directing the blocks to respective caches in different pages of the same physical memory.

D. The Mattson, Jr. Patent Does Not Disclose All Of The Limitations Of Claim 34

The Examiner impliedly admits that the Mattson, Jr. patent does not disclose the claimed marking of selected portions of the information that are to be cached according to cacheability determinations by asserting that this limitation is "equivalent" to marking branch instructions as either strong or weak as taught by Mattson, Jr. However, to support an anticipation rejection, a prior art reference must show all of the limitations of a rejected claim. Anticipation cannot be established by equivalency. Furthermore, branch prediction is not equivalent to caching for the reasons explained above. For example, caching can be performed on all instructions as well as data rather than just on branch instructions, and it involved storing cached information in a separate memory rather than simply in different locations of the same memory.

The Mattson, Jr. patent also fails to disclose the execution steps of claim 34. In particular, the Mattson, Jr. patent does not disclose detecting the marking of the selected portions of information and directing the selected portions of the information to the cache circuitry according to the marking. The teaching of Mattson, Jr. could be considered to teach this subject matter only if the substantial differences between caching and branch prediction were entirely ignored, and the substantial differences between storing cached instructions in a separate cache memory and storing branching instructions in different locations in the same memory were also ignored. Yet it is well settled that an anticipation rejection cannot be made by ignoring claim limitations that are absent from the allegedly anticipatory reference. The anticipation rejection of claim 34 should therefore be reversed.

3. Claim 48 Is Not Anticipated By U.S. Patent No. 6,115,809 to Mattson, Jr.

A. The Subject Matter Of Claim 48

Claim 48 reads as follows:

> 48. A method for compiling a computer program, comprising:
>
> making cacheability determinations for information associated with the computer program; and
>
> marking at least selected portions of the information according to the determinations.

Claim 48 is therefore directed to the compiling steps that were included in claim 34.

B. The Subject Matter Disclosed in the Mattson, Jr. Patent

The subject matter of the Mattson, Jr. patent is described in detail above. In the interest of brevity, that explanation will not be repeated here.

C. Summary Of The Rejection

The final rejection dated June 6, 2005 also rejected claim 48 as being anticipated by the patent to Mattson, Jr. To make the rejection, the Examiner relied on the same subject matter of the Mattson, Jr. patent relied on for the anticipation rejection of claim 34. Again, no explanation was give as to why teaching relating to branch prediction were applicable to instruction and data caching.

D. The Mattson, Jr. Patent Does Not Disclose All Of The Limitations Of Claim 48

As explained above with reference to claim 34, the Mattson, Jr. patent does not teach the compiling steps of claim 48, which were included in claim 34. Specifically, the Mattson, Jr. patent does not teach making cacheability determinations nor does it teach marking at least selected portions of the information according to the determinations. Instead, it teaches making branching behavior determinations and marking the computer program according to the branching behavior determinations so that the branch instructions can be processed by either static or dynamic procedures. It is not proper to simply ignore the substantial differences between caching and branch prediction to support an anticipation rejection. The anticipation rejection of claim 48 should therefore be reversed.

4. Claim 1 Would Not Have Been Obvious Over U.S. Patent No. 6,115,809 In View Of U.S. Patent No. 5,765,037 To Morrison

A. The Subject Matter Of Claim 1

Claim 1 reads as follows:

> 1. A computer system having cache circuitry, the computer system adapted to be controlled by a computer program to cache information, comprising:
>
> cache circuitry, including a cache memory adapted to store information related to a computer program;
>
> a main memory adapted to store the information;
>
> a processor adapted to be controlled by the computer program and adapted to cooperate with a bus interface unit to direct selected portions of the information to the cache circuitry based at least in part on cacheability determinations made during compilation of the computer program; and
>
> bus circuitry, operatively connecting the processor, the cache circuitry, and the main memory.

Claim 1 is therefore directed to a computer system having both main memory and cache circuitry including a cache memory. The system also includes a processor connected to the main memory and cache circuitry through bus circuitry. The processor is described as directing selected portions of the information stored in the main memory to the cache circuitry based at least in part on cacheability determinations made during compilation of the computer program.

B. The Subject Matter Disclosed in the Mattson, Jr. Patent

The subject matter of the Mattson, Jr. patent is described in detail above. In the interest of brevity, that explanation will not be repeated here.

C. The Subject Matter Disclosed in the Morrison Patent

The Morrison patent describes a technique for more efficiently executing branch instructions in a parallel processing computer system by re-ordering the instructions. The Morrison patent teaches analyzing the instructions in a program during compilation based on the manner in which the instruction uses resources, such as whether the instructions are independent and can therefore be executed in parallel. Each instruction is then assigned an execution time, *i.e.*, the amount of time required to execute the instruction. This execution time is referred to as the "instruction firing time" or "IFT." The instruction is also assigned other data, such as the identity of one of several processors that will execute the instruction. Finally, execution sets of instructions are built by re-ordering the instructions based upon the instruction firing time, the identity of the processor executing the instruction, etc.

D. Summary Of The Rejection

The patent to Mattson, Jr. was cited for disclosing essentially all of the elements of claim 1 except for the bus interface unit. The patent to Morrison was cited for disclosing this limitation. The Office Action also cited the patent to Morrison for disclosing "a system to route instructions data to the appropriate hardware, e.g., cache, using compilation information (see Morrison: col. 33, lines 16-23)." According to the Examiner:

> it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement a bus interface unit as taught by Morrison to Mattson's system, in case the later does not readily include one such unit, because this would allow the intermediate step of spatially and logically re-directing or selectively dispatching of data to the correct storage place, cache or buffers, enhancing fault-free transmission of cashable data via bus (Morrison: col. 30, lines 4-17).

The Morrison patent was not cited for disclosing a separate main memory from which selected instructions are fetched for storage in cache memory based on decisions made during compilation that, as explained above, is missing from the teachings of Mattson, Jr.

E. The Mattson, Jr. Patent In Combination With The Morrison Patent Does Not Suggest The Subject Matter Of Claim 1

The use in the Mattson, Jr. system of Morrison's bus interface unit would not result in the subject matter of claim 1. As described in the Abstract, the Mattson, Jr. patent teaches that "dynamic and static code caches are defined in physical memory by allocating pools of memory pages that have prediction flag set to dynamic and static respectively." The blocks of instructions associated with each of the branch instructions are then stored in the same page of physical memory. A processor fetches and then executes instructions from the same page of physical memory. Thus, all of the instructions are fetched from a single memory, albeit from different pages of that memory. The patent does not disclose fetching and then executing some instructions from a system memory and other some instructions from a cache memory. In fact, as explained above, the Mattson, Jr. patent does not even disclose a main memory and a separate cache memory. Therefore, the teachings of Mattson, Jr. combined with the teachings of Morrison would not result in a computer system having separate main memory and cache memory.

The Mattson, Jr. patent in combination with the Morrison patent also fails to disclose a system that makes cacheability determinations, as opposed to branch prediction determinations. As explained above, the concept of making branch prediction determinations is markedly different from the concept of making cache determinations. For example, caching can be performed on all instructions as well as data rather than just on branch instructions. Caching involves storing cached information in a separate memory rather than simply in different locations of the same memory. Caching can be performed in computer systems that do not perform instruction prefetching whereas branch prediction is useful only in prefetching computer systems.

The Office Action cites the patent to Morrison for disclosing "a system to route instructions data to the appropriate hardware, e.g., cache, using compilation information (see

Morrison: col. 33, lines 16-23)." The portion of Morrison cited in the above quote reads as follows:

> The additional and required information comprises two components, a static and a dynamic component; and the information is termed 'shared context storage mapping' (SCSM). The static information results form the compiler output and the TOLL software gleans the information from the compiler generated instruction stream and attaches the register information to the instruction prior to its being received by an LRD.

However, the "additional and required information" referred to has nothing to do with information relating to the cacheability of information stored in main memory. Instead, as described in the preceding paragraph, "[a]s several users may be executing in the processor elements at the same time, *additional pieces of information* must be attached to each instruction prior to its execution to uniquely identify the instruction source and any resources that it may use." [Col. 33, lines 6-10, emphasis added]. Thus, the "additional and required information" attached to each instruction serves the purpose of identifying the program containing that instruction since several different programs may be executed simultaneously in the disclosed multi-processor system. The "additional and required information" does not even remotely relate to the cacheability of instructions.

Although not discussed in the Office Action, the system disclosed in the Morrison patent does include a conventional caching system. However, in contrast to storing information in cache memory based on cacheability determinations made at compilation as in applicant's system, the determination of what to store in the caches of the Morrison system are made during execution of the program. Specifically,

> The execution set hardware operates independently of instruction fetching to control the movement of instruction words from main memory to the instruction cache. This hardware is responsible for fetching basic blocks of instructions into the cache until either the entire execution set resides in cache or program execution has reached a point that a branch has occurred to a basic block outside the execution set.

[Col. 24, lines 53-60]. Thus, when a processor requests an instruction from main memory, the hardware transfers to the instruction cache the entire set of instructions in that same execution set. As explained above, the execution set contains the re-ordered instructions. The decision to cache instructions has nothing to do with cacheability determinations made during compilation. Instead, all of the instructions are cached whenever the execution set in which they are contained is executed. Therefore, to the extent that the Morrison patent contains any teaching that are relevant to claim 1, it teaches away from the subject matter of claim 1.

Data is stored in the data cache of the Morrison system in essentially the same manner that instructions are stored in the instruction cache. Specifically, when a processing element attempts to read data, "[i]f the requested datum is not present in the data cache, the address delivered to the cache 1592 is sent to the data cache ATU 1580 to be translated into a system address. The system address is then issued to memory. In response, a block of data from memory (a cache line or block) is delivered into the cache partition circuits 1592 under control of data cache control 1586." [Col. 28, lines 50-56].

The Morrison patent thus fails to disclose the subject matter of claim 1 that is missing from the teachings of the Mattson, Jr. patent, *i.e.*, making cacheability determinations during compilation and using such cacheability determinations as a basis for determining which portions of data or instructions stored in the main memory should be stored in cache memory.

F. Neither The Mattson, Jr. Patent Nor The Morrison Patent Teach Any Motivation For Combining Their Teachings

It is well settled that it is not proper to combine the teachings of two or more references to support an obviousness rejection in the absence of some teaching in either reference of the desirability of combining their respective teachings. The Office Action does not cite any portion of either patent that provides this teaching. Moreover, it does not seem possible for the respective teachings of the references to be combined even in hindsight since they each use distinctly different techniques to more efficiently execute computer programs, *i.e.*, the Mattson, Jr. system executes instructions using branch prediction and the Morrison system executes instruction that have been re-ordered. In fact, the Morrison patent specifically teaches *not* caching instructions when a branch instruction is encountered (See, above quote, basic blocks of instructions a fetched into the cache "until ... a branch has occurred").

In summary, the teachings of the cited references cannot properly be combined, but even if they were combined, they still would not teach a computer system that stores in cache memory portions of the information stored in the main memory based on cacheability determinations that were made during compilation of a computer program being executed by the computer system. The obviousness rejection of claim 1 should therefore be reversed.

VIII. APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

Attached hereto is a copy of pending claims 1-61, which are involved in this appeal.

Respectfully submitted,

DORSEY & WHITNEY LLP

Edward W Bulchis

Edward W. Bulchis
Registration No. 26,847

EWB:dms

Enclosures:
Postcard
Check
Transmittal Letter (+ copy)
Two copies of this Brief

1420 Fifth Avenue, Suite 3400
Seattle, WA 98101
Tel: (206) 903-8800
Fax: (206) 903-8820

APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

1. A computer system having cache circuitry, the computer system adapted to be controlled by a computer program to cache information, comprising:

cache circuitry, including a cache memory adapted to store information related to a computer program;.

a main memory adapted to store the information;

a processor adapted to be controlled by the computer program and adapted to cooperate with a bus interface unit to direct selected portions of the information to the cache circuitry based at least in part on cacheability determinations made during compilation of the computer program; and

bus circuitry, operatively connecting the processor, the cache circuitry, and the main memory.

2. The system of claim 1, wherein the information comprises instructions of the computer program.

3. The system of claim 1, wherein the information comprises data accessed by the computer program.

4. The system of claim 1, wherein the selected portions are marked by a compiler during the compilation of the computer program such that the bus interface unit can identify the selected portions during execution of the computer program.

5. The system of claim 1, wherein each piece of the information contains marking bits, and a compiler sets the marking bits of the selected portions of the information during the compilation of the computer program.

6. The system of claim 1, wherein the compilation of the computer program comprises translating a source code of the computer program to an object code.

7. The system of claim 1, wherein the compilation of the computer program comprises programming an object code for the computer program directly.

8. The system of claim 1, wherein the cacheability determinations comprise determinations that the selected portions are cacheable.

9. The system of claim 1, wherein the cache circuitry includes at least a first cache memory and a second cache memory, and wherein the cacheability determinations comprise determinations as to whether to cache each of the selected portions in the first or second cache memory.

10. The system of claim 9, wherein the first cache is a level-one cache and the second cache is a level-two cache.

11. The system of claim 1, wherein the cache circuitry supports both write-back and write-through caching methods, and the cacheability determinations comprise determinations whether each of the selected portions is cacheable using the write-back or write-through caching method.

12. The system of claim 1, wherein the cache circuitry includes at least one N-way associative cache, wherein N is a number greater than one.

13. The system of claim 1, further comprising a compiler adapted to optimize the cacheability determinations.

14. The system of claim 13, wherein the compiler is adapted to optimize the cacheability determination for a first piece of the information based at least in part on whether caching of the first piece of information is likely to cause thrashing of the cache circuitry.

15. The system of claim 13, wherein the cache circuitry employs a cache-management scheme, and wherein the compiler is adapted to optimize the cacheability determination for a first piece of the information based at least in part on the cache-management scheme.

16. The system of claim 15, wherein the cache management scheme comprises the level of associativity of the cache memory.

17. The system of claim 13, wherein the compiler is adapted to optimize the cacheability determination for a first piece of the information based at least in part on the likely frequency that the first piece of information will be accessed by the processor during execution of the computer program.

18. The system of claim 13, wherein the compiler is adapted to optimize the cacheability determination for a first piece of the information based at least in part on what other

piece of the information is likely to be overwritten in the cache circuitry if the first piece of information is cached.

19. The system of claim 13, wherein the cacheability determinations are accomplished during the compilation of a program code into an object code by utilizing profile-based optimizations.

20. The system of claim 1, wherein the system further comprises a system controller, adapted to retrieve and send instructions and data to the main memory via the bus circuitry.

21. The system of claim 1, wherein the system further comprises at least one bus device connecting an external storage device to the bus circuitry.

22. The system of claim 21, wherein the external storage device provides instructions utilized by the processor in performing a desired task, the instructions being optimized for cacheability.

23. The system of claim 22, wherein the instructions are compiled by a compiler adapted to optimize cacheability determinations.

24. The system of claim 1, wherein the cache circuitry and the processor are provided on a single chip.

25. A system for determining which portions of a program code to cache and which to not cache, comprising:

a memory device containing a program code; and

a processor connected to the memory device, the processor being adapted to be controlled by the program code to direct selected portions of the program code to a cache based at least in part on cacheability determinations made during compilation of a computer program.

26. The system of claim 25, wherein the program code includes instructions.

27. The system of claim 25, wherein the program codes includes data.

28. The system of claim 25, wherein each of the selected portions of the program code contains at least one marking bit designating the selected portion as suitable for caching, the marking bit being set by a compiler during compilation of the computer program.

29. The system of claim 25, the processor is connected to the memory device via bus circuitry.

30. The system of claim 25, wherein the processor further comprises a level one cache.

31. The system of claim 30, wherein the system further comprises a level two cache connected to the processor and the memory device via a bus circuitry.

32. The system of claim 25, wherein the memory device further comprises a main memory for a computer system.

33. The system of claim 25, wherein the memory device further comprises an external storage device connected to and accessible by the processor via a bus circuitry.

34. A method for controlling the cacheability of information in a computer system, comprising:

compiling a computer program, by:

making cacheability determinations for information associated with the computer program; and

marking at least selected portions of the information according to the determinations;

executing the computer program on a computer system, the computer system including cache circuitry;

detecting the marking of the selected portions of the information during execution of the computer program; and

directing the selected portions of the information to the cache circuitry according to the marking.

35. The method of claim 34, wherein the information comprises instructions of the computer program.

36. The method of claim 34, wherein the information comprises data to be accessed by the computer program.

37. The method of claim 34, wherein each piece of the information contains marking bits and the act of marking includes setting the marking bits of at least the selected portions of the information.

38. The method of claim 34, wherein the step of compiling comprises translating a source code of the computer program to an object code.

39. The method of claim 34, wherein the step of compiling comprises programming an object code for the computer program directly.

40. The method of claim 34, wherein the act of making cacheability determinations comprises determining that the selected portions are cacheable.

41. The method of claim 34, wherein the cache circuitry comprises a first cache memory and a second cache memory and wherein the act of making cacheability determinations comprises determining whether to cache each of the selected portions in the first cache memory or the second cache memory.

42. The method of claim 34, wherein the cache circuitry supports both write-back and write-through caching methods, and the act of making cacheability determinations comprises determining whether to cache each of the selected portions using the write-back or write-through caching method.

43. The method of claim 34, wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on whether the caching the first piece of information is likely to cause thrashing of the cache circuitry.

44. The method of claim 34, wherein the cache circuitry employs a cache-management scheme, and wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on the cache-management scheme.

45. The method of claims 34, wherein the cache circuitry includes at least a first cache memory, and wherein the act of making cacheability determinations for information associated with the computer program comprises making cacheability determinations based on the level of associativity of the first cache memory.

46. The method of claim 34, wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on the likely frequency that the first piece of information will be accessed by the processor during execution of the computer program.

47. The method of claim 34, wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on what other piece of the information is likely to be overwritten in the cache circuitry if the first piece of information is cached.

48. A method for compiling a computer program, comprising:

making cacheability determinations for information associated with the computer program; and

marking at least selected portions of the information according to the determinations.

49. The method of claim 48, wherein the information comprises instructions of the computer program.

50. The method of claim 48, wherein the information comprises data to be accessed by the computer program.

51. The method of claim 48, wherein each piece of the information contains marking bits and the act of marking includes setting the marking bits of at least the selected portions of the information.

52. The method of claim 48, further comprising translating a source code of the computer program to an object code.

53. The method of claim 48, further comprising programming an object code for the computer program directly.

54. The method of claim 48, wherein the act of making cacheability determinations comprises determining that the selected portions are cacheable.

55. The method of claim 48, wherein the act of making cacheability determinations comprises determining whether to cache each of the selected portions in a first cache memory or a second cache memory.

56. The method of claim 48, wherein the act of making cacheability determinations comprises determining whether to cache each of the selected portions using a write-back or a write-through caching method.

57. The method of claim 48, wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on whether the caching the first piece of information is likely to cause thrashing of the cache circuitry.

58. The method of claim 48, wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on a cache-management scheme.

59. The method of claim 58, wherein the cache management scheme comprises the level of associativity of a cache memory.

60. The method of claim 48, wherein the act of making cacheability determinations includes making a cacheability determination for a first piece of the information based at least in part on the likely frequency that the first piece of information will be accessed by a processor during execution of the computer program.

61. The method of claim 48, wherein the act of making cacheability determinations comprises making a cacheability determination for a first piece of the information based at least in part on what other piece of the information is likely to be overwritten in a cache circuitry if the first piece of information is cached.